Predicting Anxiety Levels with Machine Learning
A White Paper on Model Development and Potential Applications

Konstantin Zuev
Department of Computer Science
Dalhousie University
Halifax, NS, Canada
kn905954@dal.ca

Mohammed Usama Jasnak
Department of Computer Science
Dalhousie University
Halifax, NS, Canada
mh659974@dal.ca

Abstract

Anxiety disorders affect many youths and adults, but traditional methods of assessing anxiety rely on subjective self-reports, which can be inconsistent. This study aims to develop a machine learning model that predicts anxiety levels using mobile sensing data. Data was collected over 14 days from 161 participants using the PROSIT app, including anxiety scores from SCARED and CESD assessments. The sensor data tracked mobility, sociability, screen usage, physical activity, and sleep patterns. Leveraging synthetic data generated with Conditional Generative Adversarial Networks, bootstrapping, and balanced loss functions we were able to provide accurate anxiety assessments. Moreover, we discussed the importance of ensuring that a model provides a good estimate of the probability of class membership under operational conditions, i.e. the posterior probability, which is important because it enables reliable decision-making by reflecting the true likelihood of outcomes.

1 Introduction

Anxiety disorders are the most common mental health conditions, affecting up to 33.7% of the population over a lifetime [1]. These disorders lead to major healthcare costs and can take a significant toll on one's health. However, current methods for assessing anxiety often rely on subjective reports, which can be unreliable. There is a need for more objective tools to evaluate anxiety accurately.

Mobile sensing technologies, which track daily behaviours like movement, social interactions, and sleep, offer a new way to assess anxiety. With advancements in artificial intelligence (AI), it is now possible to use these data to predict anxiety levels.

This study aims to develop a machine learning model to predict anxiety using mobile sensing data. Conducting such studies is naturally costly and challenging. Therefore, we aim to explore maximizing the use of limited data by leveraging data augmentation approaches, synthesizing additional data and employing various techniques to prevent overfitting.

2 Literature review

Using machine learning to detect anxiety disorders offers several key advantages. Firstly, Machine Learning (ML) algorithms can integrate data from various sources, including medical records, demographic details and sensor data, to uncover patterns that traditional methods might miss. This makes it possible to achieve more accurate and timely diagnoses, even when symptoms are subtle. Additionally, ML models can analyse information from unconventional sources, such as wearable devices, mobile apps and social media, enabling continuous and non-invasive mental health monitoring. This approach helps in detecting early signs of anxiety, allowing for quicker intervention and

potentially preventing the condition from escalating. Additionally, ML-based approaches also help reduce dependence on subjective assessments by offering a more objective, data-driven foundation for diagnosis. By automating the detection process, these systems can reduce the workload of healthcare professionals, giving them more time to focus on direct patient care.

As a result, there have been numerous attempts to utilise ML models for detecting anxiety and other mental health disorders. As early as 2016, ensemble trees were used to predict the persistence and severity of major depressive disorder, outperforming previously used conventional methods [2]. Another example of a successful application of ML methods for screening mental health illnesses is a paper by Arkaprabha Sau and Ishita Bhakta [3]. The authors used socio-demographic and occupational factors, as well as comorbid conditions, to build multiple ML models, including Random Forest and Catboost. They reported achieving high levels of accuracy, precision, and recall for a relatively small dataset of 470 participants.

2.1 Small data problem

One frequent challenge of applying ML to healthcare problems is the lack of data. In many cases, datasets have only a few hundred participants, making it difficult to leverage even some traditional (shallow) machine learning algorithms. For instance, applications for depression had thus far relied on small samples and limited predictor sets, which failed to tap into the full potential of the methods [2]. Therefore, it is necessary to build a model that is complex enough to capture the necessary patterns and, at the same time, make sure that it will not overfit.

Aside from collecting more data, which is not always feasible, one can focus on increasing the amount and diversity of the available data. There are various methods to achieve this such as by generating more data. The simplest method is bootstrapping, a popular technique for increasing sample size in methods like Random Forest [4]. While bootstrapping increases data diversity, it does not extend significantly beyond the initial data distribution, which may be insufficient for building a good model as we have to rely on a small dataset being representative enough. In such case, generating synthetic data using Conditional Generative Adversarial Networks (CTGAN) can be another solution [5]. According to the paper, CTGAN outperforms existing deep learning models like MedGAN [6] and VeeGAN [7], as well as Bayesian methods [8, 9]. This improvement is achieved by using mode-specific normalisation to handle non-Gaussian and multimodal distributions, along with conditional generation and training-by-sampling to condition the generator on discrete column values, which addresses the imbalance issues in categorical data.

Similar findings were reported in a systematic review of synthetic data generation methods, where GAN-based approaches performed well in generating high-quality medical data [10].

2.2 Class Imbalance

Imbalanced data is a common issue in medical datasets. This imbalance occurs because certain medical conditions naturally have fewer instances compared to more common conditions. Even datasets that do not have extreme imbalances, let's say a ratio of 1:4, can make it difficult for machine learning models to learn from and accurately predict minority classes.

Addressing this issue requires careful consideration of techniques suitable for tackling this imbalance. The authors of "A survey on imbalanced learning: latest research, applications and future directions" [11] and "A broad review on class imbalance learning techniques" [12] experimented with various strategies at different levels to handle imbalanced classes. The main strategies include data pre-processing techniques, algorithm modifications, and hybrid approaches. Data-level approaches involve modifying the training dataset to balance the class distribution, either by oversampling the minority class or undersampling the majority class. Common techniques include random oversampling and the Synthetic Minority Over-sampling Technique (SMOTE). Algorithm-level approaches are modifications made to the learning algorithms, i.e. cost-sensitive learning. Lastly, hybrid methods combine data-level and algorithm-level strategies, often integrating ensemble techniques to improve overall performance. Examples include SMOTE combined with boosting.

However, given that our dataset is quite small, we also aim to incorporate synthetic data generation techniques, as these methods can help augment the training set, providing the model with more diverse examples.

2.3 Deep learning models for small tabular datasets

Traditional machine learning techniques frequently used for classification, such as XGBoost or Random Forest, are often favored for their simplicity, interpretability, and efficiency, particularly when working with smaller datasets. To compare their performance with deep learning, Shwartz-Ziv and Armon [13] trained and evaluated different deep learning models alongside XGBoost on 11 diverse tabular datasets. The research findings suggest that XGBoost consistently outperforms deep learning models for tabular data. Although XGBoost may excel in many scenarios and is the easiest to optimise, it may not always achieve the best performance. Therefore, it is essential to explore both traditional ML and deep learning.

Aside from traditional machine learning models, which have been shown to work well for tabular datasets, utilising fully connected neural networks can be a good choice for constructing a wide range of feature combinations that are also non-linear in nature. An extension of this approach was used to develop HyperTab, a hypernetwork-based method for solving small-sample problems on tabular datasets [14]. To tackle small-dataset challenges, the authors represented each data point as a random subset of its features, increasing the number of points from n to $n \times k$, where k is the number of augmentations. This method limited the information contained in each data point while avoiding the introduction of noise. Additionally, borrowing from ensemble-like techniques, they opted for a more parsimonious variant—instead of training m individual networks, they created a central mechanism to generate the entire ensemble, known as a hypernetwork.

According to their findings, HyperTab consistently outperformed XGBoost, Random Forest (RF), and regular multilayer perceptrons (MLPs) on various small tabular datasets. The experiments were conducted on 22 public datasets and 20 microbial datasets, focusing on classification tasks evaluated using balanced accuracy. On small datasets (fewer than 1,000 samples), HyperTab consistently ranked first with statistically significant improvements, such as scoring 95.27% on the Parkinsons dataset compared to XGBoost (86.35%) and RF (86.84%), and 70.59% on Hill-Valley without noise compared to XGBoost (65.53%) and RF (57.33%). The datasets covered diverse domains, including clinical, compositional and radar data.

To conclude, a balanced approach that considers the strengths and weaknesses of both shallow ML and Deep Learning (DL) can lead to more robust solutions across diverse data scenarios.

3 Dataset analysis

The data were collected using the PROSIT app [15], which gathers metrics such as mobile phone use, sleep, GPS data and self-reported sense of well-being. Initially, these metrics were recorded for all participants at different time intervals. For example, the number of times a phone is unlocked could be monitored throughout the day, while self-assessment tests might be taken once a week. Additionally, some participants did not grant permission to collect certain metrics, such as GPS data. Finally, only a subset of participants were professionally assessed for the presence of anxiety disorders, providing target labels for training supervised machine learning models. While there are hundreds of additional users available in the dataset, these participants were not professionally diagnosed, their data lacks the clinically verified labels required for accurate training and validation of models. Consequently, utilising this unassessed data for model training would risk introducing inaccuracies and biases.

To address the issue of irregular timestamps, the raw data were aggregated over two-week intervals for 161 participants who were monitored throughout the entire study and did not drop out. To minimise information loss after data aggregation, additional features were created. For example, beyond averaging anxiety scores, metrics like the standard deviation were calculated to capture the variability in scores over time.

After processing the raw data, a total of 322 observations (161 for each week) were selected. The dataset was split into two equal subsets of 161 observations each: the first week for training and the second week for testing. This approach was used to resemble a real-world application, where data is collected for an individual over time, and based on this information, the presence of anxiety is detected.

A more detailed breakdown of feature groups is presented in Table 1. Feature groups were formed by categorising related variables based on the type of data source. For example, the group "Drug usage" includes variables such as the use of tobacco, alcohol and cannabis in the past 90 days.

Feature group	Number of features
Drug usage	9
Anxiety (self-reported)	4
GPS data	5
Call data	4
Screen time	8
Social life	1
Sleep	4
Physical activity	11
Demographics	9
Total	**55**

Table 1: Feature groups and the corresponding number of features.

In our analysis, we excluded features with more than 25% missing values. Given the size of the dataset, imputing missing data for such variables was deemed impractical. Moreover, imputing missing values in such cases can introduce bias and distort relationships between variables. The results may not reflect real patterns, leading to misleading conclusions. Therefore, a stricter cutoff was chosen. For instance, over 40% of participants did not report their income. Consequently, three variables were excluded: "income," "average night intensity" (mobile phone light exposure at night), and "location variance" (natural logarithm of variance in GPS data).

As part of our pipeline, we considered imputing missing values using the mean for numeric variables and the mode for categorical variables. Although this method ignores relationships with other variables, it can serve as a baseline, especially for very small datasets. Another popular method for missing data imputation is MICE [16], which provides a more sophisticated imputation by accounting for variable relationships. The effectiveness of each approach will be evaluated based on out-of-sample error. Recent years have seen increased use of DL-based methods for handling missing values, showing improved imputation accuracy [17, 18]. However, Sun, Li, Xu, Zhang, and Wang [19] observed that deep generative models exhibited instability and uncertain convergence when applied to datasets of small, moderate, or even relatively large sizes, particularly when the sample size was under 30,000. These methods require a large amount of data for training, which is not feasible in our case.

Another challenge we faced was data imbalance. The ratio of patients with anxiety to healthy individuals is roughly 1 to 4. Since the dataset is small, relying on data augmentation techniques such as oversampling the minority class can be very challenging—there is not enough variance in the data to properly represent the minority class, with only 42 observations in the training set. Therefore, we opted for algorithmic modifications, such as weighted cross-entropy or class weighting when calculating the impurity score.

The dataset contained some participants for whom GPS data was recorded incorrectly. For example, latitude and longitude coordinates were drastically changing every minute, akin to moving from one part of the world to another. Such data points were removed.

We also considered bootstrapping and the use of CGANs [5] for our data processing pipeline to enhance the quantity and diversity of the training set. We encountered two key problems. First, since the train and test sets included the same participants one week apart, overfitting on the train set could result in an artificially good performance on the test set, as confirmed during model training (discussed in the Results section 5). Second, increasing the dataset size by sampling with replacement was problematic due to the small sample size, leading to insufficient variability in the data. To address these issues, we experimented with injecting synthetic data of varying sizes generated using CTGANs. While CTGANs are designed to produce data that matches the underlying training distribution, they may introduce some noise, potentially helping to make the model more robust during training.

Ultimately, we evaluated which approach performed better by analysing the performance metrics of the downstream machine learning models. Additionally, we used the two-sample Kolmogorov–Smirnov statistic to compare the distribution of each variable in the train and test sets. The Kolmogorov–Smirnov statistic measures the maximum distance between the cumulative distribution functions of two samples and ranges from 0 to 1, where a value closer to 0 indicates more similar distributions.

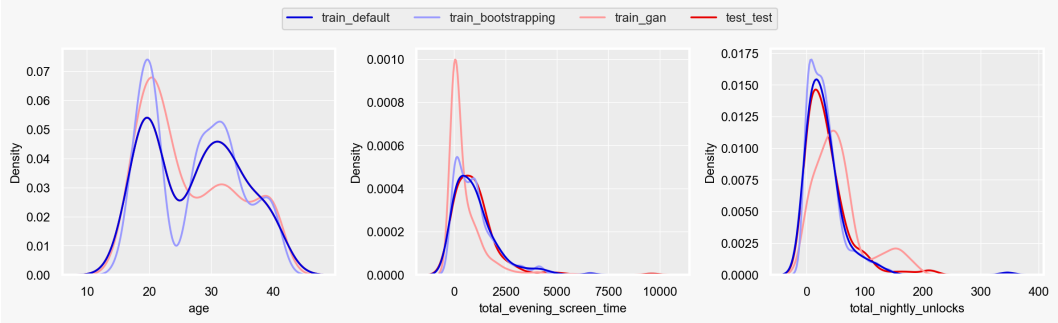


Figure 1: Feature distributions of three variables: raw, bootstrapped, CTGAN.

The distributions of data generated with CTGAN differ significantly from both bootstrapped and raw data, which can be useful for enhancing data variety.

$$D_{n,m} = \sup_x |F_n(x) - G_m(x)|, \tag{1}$$

where item $F_n(x)$ is the empirical cumulative distribution function (ECDF) of a given variable in the train set, $G_m(x)$ is the ECDF of the same variable in the test set, and $\sup$ denotes the supremum (maximum) of the absolute differences over all possible values of x.

The averages of $D_{n,m}$ over all variables for the original train set, bootstrapped train set and CTGAN train set were 0.079, 0.084 and 0.335, respectively. It can be seen that there was hardly any difference between the average of $D_{n,m}$ for the original train set and the average of $D_{n,m}$ for the bootstrapped dataset. On the other hand, the synthetic dataset created using CTGAN resulted in fairly different distributions.

These findings were also confirmed after we analysed feature distributions. Figure 1 illustrates three variables and their respective distributions after applying data augmentation techniques. It can be clearly seen that CTGAN produced distributions that were noticeably different.

We also conducted Principal Component Analysis (PCA) to identify any clusters and determine whether linear combinations of variables would explain a decent portion of the variance in the data. The first three components accounted for roughly 43.2% of the variance, with the first component explaining 18.9%. This suggests that the data might be noisy, that linear combinations may not sufficiently capture the variance, or both. The results are depicted in Figure 2. The only potentially interpretable cluster appears on the second biplot: all patients with a PC2 score higher than 2 did not have anxiety. PC2 was positively correlated with "total incoming calls," "total outgoing calls," and "total call duration," with standardised loadings of 0.46, 0.43, and 0.39, respectively. This suggests that patients who frequently communicated via phone were less likely to have anxiety.

4 Methodology

4.1 Small Sample Size

Due to their complexity, many machine learning models require larger datasets than statistical methods. Training ML models can be particularly tricky if the training set consists of only 161 observations. Therefore, we explored methods for generating additional data using bootstrapping and CTGAN.

Importantly, synthetic data was not used to address the inherent data imbalance, which was handled separately, but rather to enhance the original dataset. Moreover, the test set was not augmented, which allowed us to assess the real-world performance of our models.

As outlined in the literature review, CTGANs perform better than other popular deep learning models for tabular synthetic data generation. Since the training set is small, there is a concern about either overfitting or producing distributions identical to the original data or underfitting and generating

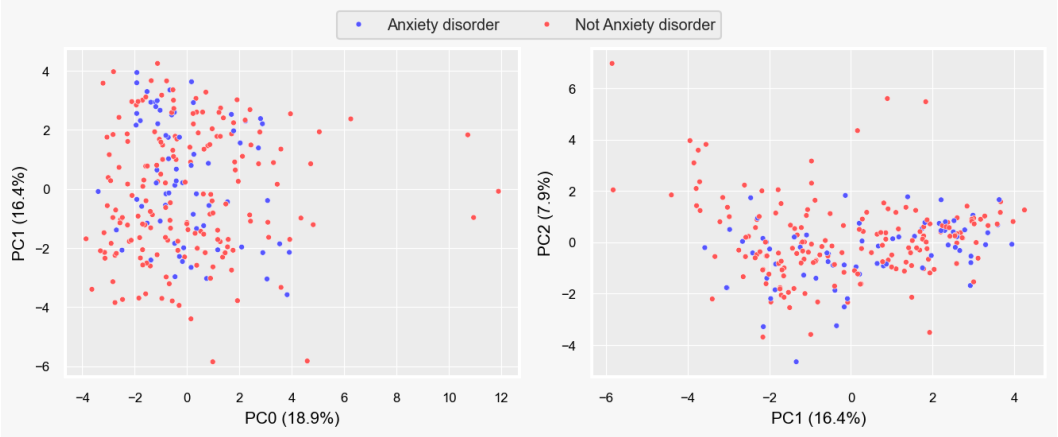


Figure 2: Principal Component Analysis. Biplots.

For this dataset, linear combinations may not sufficiently capture the variance in the data. The only discernible cluster appears among healthy patients with PC2 > 2 (second biplot), which is associated with frequent phone communication.

excessively noisy data. However, based on our analysis, neither of these issues was encountered, as demonstrated in the data analysis segment.

Since our data is not simulated, the usefulness of synthetic data was evaluated based on machine learning performance rather than likelihood fitness. We experimented with different sample sizes and investigated the effects of applying bootstrapping and CTGAN.

Furthermore, the fact that the training set is very similar to the test set may lead to good out-of-sample performance, even if the model severely overfits the data. As a result, we aimed to strike a balance between test set performance and cross-validation (CV) performance; i.e., low out-of-sample error is insufficient if cross-validation curves clearly show signs of overfitting.

4.2 Class Imbalance

For this study, we used loss functions that can inherently handle imbalanced data, such as weighted cross-entropy, instead of applying the Synthetic Minority Over-sampling Technique (SMOTE). SMOTE often falls short in handling extremely small imbalanced datasets. Primarily, using SMOTE on a small dataset can lead to overfitting, as it generates synthetic samples by interpolating between limited minority instances. This can cause the model to memorise specific patterns from these instances rather than learning generalisable trends, leading to poor performance on new data. Additionally, the lack of diversity in small datasets means that SMOTE's synthetic data closely resemble existing samples, adding redundancy rather than meaningful variety.

Another limitation is the amplification of noise. In small datasets, outliers or noisy samples in the minority class may lead to misleading synthetic samples because SMOTE does not differentiate between clean and noisy data. Furthermore, the concept of forming "neighborhoods" of samples is difficult to achieve with few data points, resulting in unreliable synthetic samples that may not capture the minority class distribution accurately.

Moreover, it has been empirically shown that when the objective metric is proper balancing could improve prediction performance for weak classifiers but not for the strong state-of-the-art (SOTA) classifiers [20].

4.3 Machine Learning Frameworks

4.3.1 Shallow Models

Random Forest (RF) and XGBoost have been extensively used because perform exceptionally well in the context of tabular data. As ensemble techniques, they can produce complex models capable of capturing non-linear patterns and feature interactions. At the same time, they incorporate several

safeguards to prevent overfitting, such as random feature sampling, control over the complexity of each weak learner, and built-in L1 and L2 regularisation in the case of XGBoost.

To combat class imbalance, `scale_pos_weight` was used for XGBoost. This parameter is defined as the sum of negative instances divided by the number of positive instances, assuming that the positive class is the minority. It directly scales errors of the positive class, forcing the model to over-correct them. In the case of RF, `class_weight == 'balanced'` was employed. It assigns weights inversely proportional to class frequencies, which affects the metric calculation during node splitting.

As a simple benchmark, We also trained Logistic Regression (LR) with L2 penalty and `class_weight == 'balanced'` that helps in handling imbalanced data by employing a weighted log-loss with inversely proportional to class frequencies.

4.3.2 Deep Learning Models

In this study, we explored three deep learning models: TabNet [21], HyperTab [14] and a modified Deep Neural Network (DNN).

TabNet uses a sequential attention mechanism to selectively focus on the most important features at each decision step. Unlike traditional models that process all features simultaneously, TabNet employs feature masks created by an attentive transformer to dynamically choose relevant features. This improves learning efficiency and interpretability. The model processes these selected features through a feature transformer, combining shared and unique layers to extract useful patterns. The use of sparsemax activation encourages a sparse selection of features, which helps in interpretability by revealing which features influenced each prediction.

HyperTab, on the other hand, employs a hypernetwork to generate weights for the target network using a binary mask that represents a subset of features. Feature subsetting provides a lower-dimensional representation of the data. When applied repeatedly, it creates a more diverse dataset, similar to the way RF benefits from random feature sampling. Moreover, this approach eliminates the need to separately train multiple smaller networks for each data augmentation, making the process more efficient while inherently incorporating ensemble-like behaviour during training. This enables HyperTab to be effective even on small datasets.

To improve HyperTab's performance on our imbalanced data, we modified the original model by adding weighted cross-entropy, which was not present in the original implementation.

The vanilla DNN was modified in the following ways:

- When developing a DNN, we incorporated ideas from HyperTab. For instance, they used feature subsetting to ensure class-invariant transformations. We applied dropout regularisation before the first hidden layer to mimic the selection of a subset of features, providing our DNN with more augmented data, in addition to using synthetic data to increase dataset size and diversity.
- We also considered taking the weighted average of the weights after training (the training loop was not modified). This approach was intended to make the final model more robust [22].
- AdamW was used as the optimizer to correctly implement L2 regularisation. This ensured that the weight decay term did not appear in the moving averages.
- Dropout layers were placed after each hidden layer, with values ranging from 0 to 0.4. This was done to introduce additional regularisation, preventing overfitting.
- Batch normalisation (BN) was used to reduce second-order relationships between parameters of different layers. Updating the parameters of a given layer can become very tricky as it depends on all the other layers, particularly when second-order terms are too large or too small [23]. Based on our experiments, introducing BN only after the first hidden layer and before the dropout layer improved the results. We hypothesise that this is because we used dropout layers after each hidden layer to combat overfitting. Therefore, it is possible that during inference, when the dropout layers are disabled, the moving means and variances of the BN layers computed after the dropout layers may be "improper" [24].

Model Training Specifics

- Each model was re-run 15 times to assess the variability of the test set error. The following variance sources were possible: random initialisation of weights, data shuffling, randomness in data augmentation, i.e. the synthetic data generate by a CTGAN.
- Random Search was used to perform hyperparameter tuning for shallow models, whereas DL models were tuned using Grid Search. Hyperparameter grids for all models are presented in Appendix A.
- All shallow ML models were implemented using `scikit-learn`, and all DL models were implemented using `PyTorch`.

4.4 Bayesian Adjustment

Normally, when for a particular disease, it is essential to account for how prior probabilities, i.e. the prevalence of anxiety, will affect True Positive Rate (TPR) and False Positive Rate (FPR). It helps us adjust metrics like TPR and FPR by taking into account the base rate of anxiety disorders in the population. This adjustment is crucial because even a highly accurate model might give misleading results. By incorporating the prevalence as a prior probability, Bayesian adjustment fine-tunes the model's output. This way, TPR and FPR better reflect true cases and reduce the risk of overestimating or underestimating diagnoses.

$$P(anxious|+) = \frac{P(+|anxious) \cdot P(anxious)}{P(+|anxious) \cdot P(anxious) + P(+|healthy) \cdot P(healthy)}, \tag{2}$$

where $+$ denotes that a test or model predicts someone as having anxiety.

Let's consider the following inputs:

- Anxiety prevalence depends on several factors, such as gender, age, geographical location and more. For instance, a systematic review on anxiety prevalence by Remes, Brayne, van der Linde, and Lafortune indicates a range between 0.04 and 0.25 [25]. For this example, we can use a more conservative estimate of 0.15 for the prior $P(anxious)$.
- For the sake of an argument, let's suppose that TPR and FPR are quite reliable: $P(+|anxious)$ 0.9 and $P(+|healthy)$ 0.1, respectively.

$$P(anxious|+) = \frac{0.9 \cdot 0.15}{0.9 \cdot 0.15 + 0.1 \cdot 0.85} \approx 0.61 \tag{3}$$

Hence, in this example, the actual probability that a person who tests positive truly has an anxiety disorder is roughly 61%. If this testing procedure were implemented on a large scale, with thousands of patients being tested, a significant number of false positives would inevitably occur. This could overwhelm healthcare professionals, further straining the resources of the healthcare system.

Consequently, we believe that improving model accuracy is not the only concern. It is also crucial to develop guidelines for selecting only those patients who are more likely to have anxiety disorders, reducing the impact of prior probability on model performance.

To some degree, ML models are more robust to this issue because, if the training set is representative of operational conditions, the classifier's output will provide a good estimate of the probability of class membership under those conditions, i.e., the posterior probability $P(anxious|+)$. This also depends on the cost function, though to a lesser degree [26]. However, this robustness is not guaranteed, particularly when the training procedure is modified to account for data imbalance. Such modifications can improve accuracy for the minority class but may result in output probabilities that are not well-calibrated. Furthermore, having a representative training set is essential yet challenging, especially in the context of small datasets, which are often supplemented with synthetic data.

In our view, it is crucial to keep these considerations in mind, even if a given model results in a low out-of-sample error. For example, while being an accurate classifier, Random Forest (RF) is known to produce poorly calibrated probabilities, which can mislead downstream decision-making

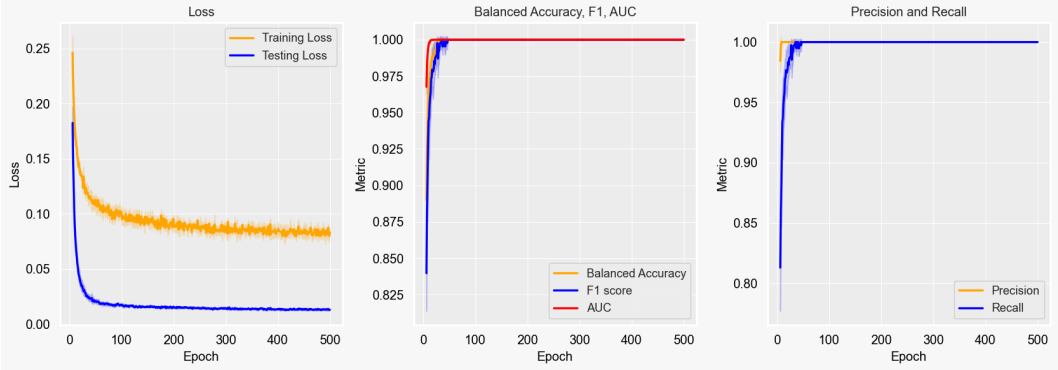


Figure 3: Cross-validation results for DNN using bootstrapping.

The aggregated results from 5-fold cross-validation for the DNN model with bootstrapping clearly demonstrate rapid overfitting.

processes, especially when these probabilities are interpreted as confidence scores. This limitation becomes particularly problematic in healthcare applications, where decisions often depend on reliable probabilistic estimates rather than simple binary predictions. Needless to say, other models can also be susceptible to this issue for a number of reasons, and it is important to ensure that this problem is addressed. The same is true for ensuring that the model has learned the priors, which will normally manifest in high minority class accuracy without a significant sacrifice to the majority class.

5 Results

5.1 Model Development Using All Variables

As discussed earlier, to ensure that our models are accurate without overfitting, considering the small sample size and the similarity between training and test sets (same participants one week apart), we compared outcomes based on cross-validation error, out-of-sample accuracy, and evidence of overfitting.

Figure 3 shows the 5-fold cross-validation error after applying DNN with bootstrapping only. The results in the figure are aggregated to display the average performance and standard deviations. In addition to the training and validation curves for the weighted cross-entropy loss, various metrics were calculated on the validation folds are also plotted.

The validation set metrics such as balanced accuracy, F1 score, AUC, precision, and recall reach their maximum values, clearly indicating overfitting. However, this did not translate into significantly worse test set performance on the real, non-augmented data. For instance, the macro F1 scores for the training and test sets were 1 and 0.877, respectively, suggesting a significant overlap between the training and test data.

In contrast, after adding synthetic data generated with CTGAN in addition to bootstrapping, the CV results improved, as shown in Figure 4. We experimented with various synthetic sample sizes, ranging from 500 to 1500, and selected 1200, as it led to more challenging training conditions, based on the CV curves—better test set performance, while keeping the difference between test and CV errors minimal. Unlike the results with bootstrapping alone, the macro F1 scores for the training and test sets were 0.806 and 0.9, respectively. For this reason, we used 1200 samples generated by CTGAN for each model in our training pipeline, except for HyperTab which has built-in mechanisms of data augmentation.

In order to assess the efficacy of different models, we aggregated the test set performance over 15 runs for each model, reporting average scores and standard deviations, as outlined in Table 2. The top score for each metric is denoted in bold, and the second-best score is underscored.

It can be seen that the proposed DNN architecture consistently outperforms other models, including its variants: DNN-MA, which uses the exponential moving average of weights, and DNNnoReg, which does not include a dropout layer before the input layer to mimic feature dropping.

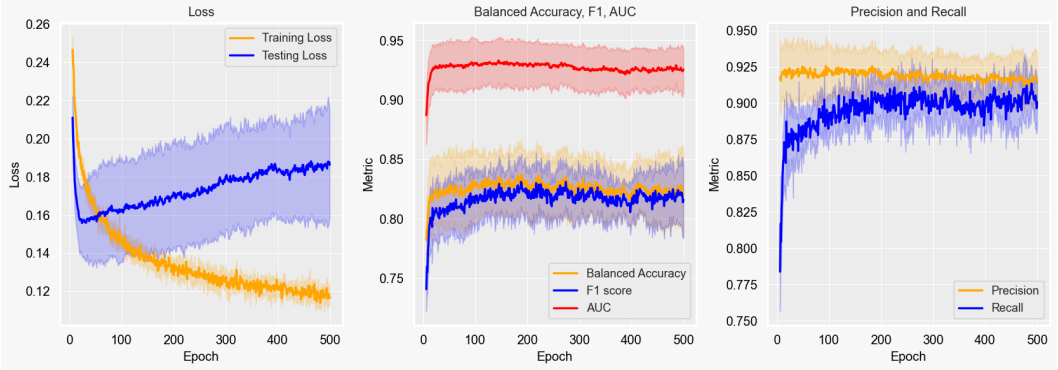


Figure 4: Cross-validation results for DNN with bootstrapping and synthetic data from CTGAN. Compared to using only bootstrapping (Figure 3), injecting synthetic data generated by a CTGAN clearly helps mitigate overfitting.

Model	Score	F1 Class 0	F1 Class 1	F1 Macro	Balanced Accuracy	AUC
LR	avg	0.641	0.829	0.735	0.775	0.852
	std	0.024	0.013	0.018	0.021	0.008
RF	avg	0.644	0.898	0.771	0.747	0.900
	std	0.064	0.020	0.041	0.037	0.045
XGB	avg	0.663	0.901	0.782	0.760	0.906
	std	0.072	0.020	0.045	0.047	0.020
TabNet	avg	0.718	0.904	0.811	0.805	0.878
	std	0.094	0.035	0.064	0.064	0.068
HyperTab	avg	0.825	0.945	0.885	0.866	0.960
	std	0.039	0.010	0.024	0.032	0.008
DNN	avg	**0.854**	**0.947**	**0.900**	**0.903**	**0.968**
	std	0.027	0.010	0.018	0.017	0.007
DNN-MA	avg	0.820	0.931	0.875	0.889	0.957
	std	0.022	0.008	0.015	0.017	0.007
DNNnoReg	avg	0.794	0.932	0.863	0.850	0.936
	std	0.035	0.012	0.023	0.024	0.015

Table 2: Aggregated model performance scores over 15 runs (all 55 variables).

Class 0 denotes the minority class, representing instances of anxiety. Unlike DNN, DNNnoReg does not include a dropout layer before the input layer. DNN-MA utilizes the moving average of weights.

As expected, DNN-MA exhibited lower variance compared to DNN, as the exponential moving average of weights reduced weight oscillations. However, its average performance metrics were worse. We compared the macro F1 and the F1 score of class 0 (the minority class) for both models using a two-sample t-test with $\alpha = 0.05$. For both metrics, the p-value was smaller than 0.001, confirming that DNN was the better model.

Interestingly, both RF and XGB failed to account for data imbalance, even with class weights, whereas all DL models, with the help of weighted cross-entropy, performed better in handling the minority class.

In the methodology section, we discussed the importance of model outputs reflecting the true distribution of the target variable. One way to assess this is by plotting the calibration curve and the label distribution, both of which are shown in Figure 5.

It is evident that the best DL model, DNN, outputs probability scores that are relatively well-calibrated, i.e. they are not consistently over- or under-estimated. This is important as we can trust

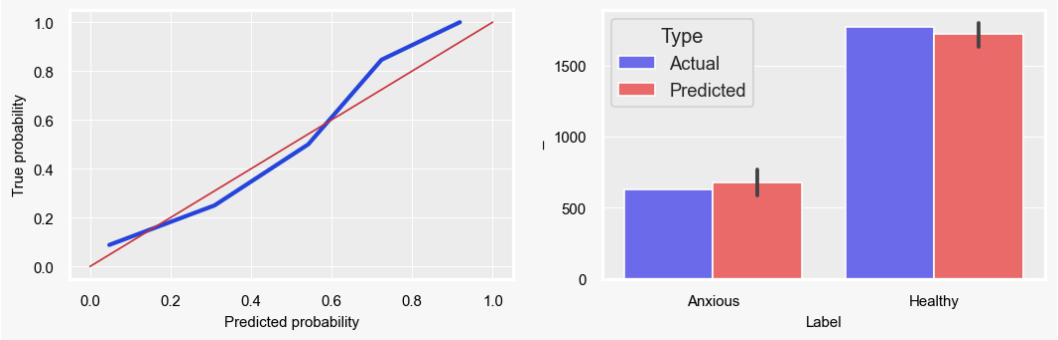


Figure 5: Calibration curve and label distribution on the test set.

The calibration curve on the left demonstrates a good match between predicted probabilities and actual outcomes, suggesting that the model's probability estimates are trustworthy. The figure on the right shows that the distribution of true and predicted labels closely matches, indicating the model is well-aligned with the true class proportions. The model is rerun 15 times to plot the confidence intervals.

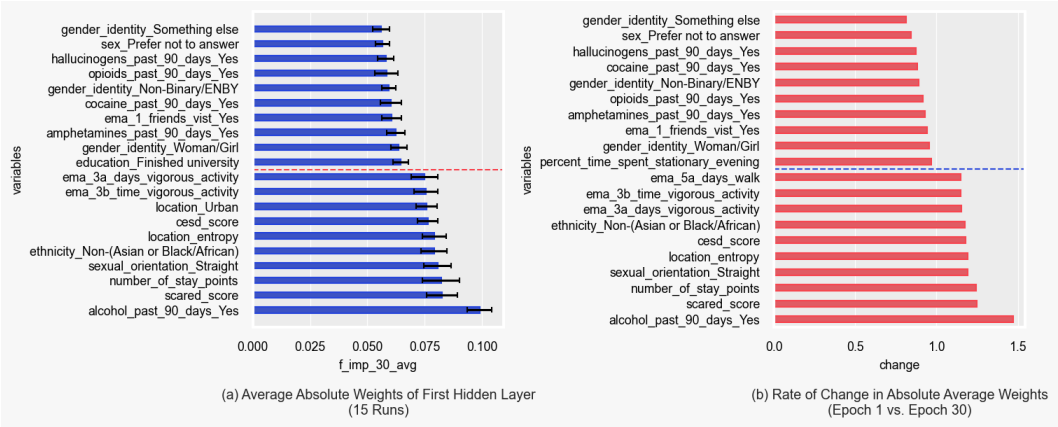


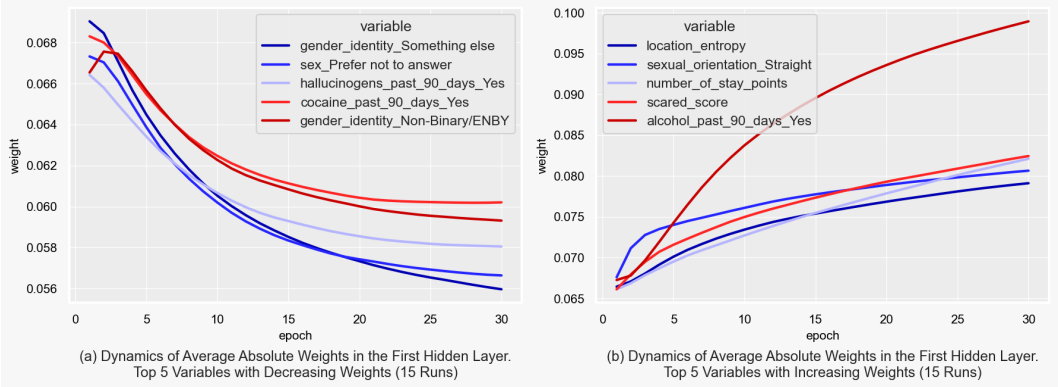
Figure 6: Analysis of Weights in First Hidden Layer.

Figure (a) depicts 20 variables with their respective lowest and highest average absolute weights at the final epoch. The model is rerun 15 times to generate the confidence intervals. Figure (b) shows 20 variables with their respective lowest and highest rates of change from epoch 1 to epoch 30.

these probabilities to reflect the true distribution of the target. Moreover, it can be seen that the distribution of true labels and predicted labels is basically the same, which means that the model is well-aligned with the true label distribution in terms of class proportions. Thus, we can assume that the posterior probability will not get significantly lower, making the model less useful. Proper probabilities suggest that the model has effectively incorporated the prior, which is crucial in case of substantial class imbalance. It also indicates that the classifier is less reliant on a specific classification threshold.

Although we did not focus on model interpretability, exploring the weights of the best-performing model may provide some insights. We analysed the weights in the first hidden layer of the DNN, loosely interpreting them as some form of feature importance. However, it should be noted that feature importance in early layers is influenced by subsequent layers. Therefore, interpreting the weights of the first layer in isolation should be approached with caution. Moreover, neural networks leverage nonlinear activation functions, making feature importance far more complex than raw weights suggest. On the other hand, in the context of tabular datasets, we have a direct mapping of features to their respective weights in the first layer, which can provide useful information.

We calculated the average of the absolute weights (AAW) in the first layer over 15 runs. Absolute values were used to focus solely on the magnitude, under the assumption that the larger weights indicate greater importance. In Figure 6, a total of 20 variables are depicted: 10 with the lowest



(a) Dynamics of Average Absolute Weights in the First Hidden Layer. Top 5 Variables with Decreasing Weights (15 Runs)

(b) Dynamics of Average Absolute Weights in the First Hidden Layer. Top 5 Variables with Increasing Weights (15 Runs)

Figure 7: Weights Dynamics in First Hidden Layer.

Variables with weights that have dramatically decreased over time may suggest they are less important. In contrast, variables with weights that have significantly increased might be more informative.

weights and 10 with the highest (Figure 6a). In general, variables associated with physical activity had higher weights, indicating greater importance.

We also computed how significantly the weights changed by comparing their values at epoch 1 with the fully trained weights at epoch 30. Figure 6b shows 10 variables with the highest growth and 10 variables with the largest decrease, defined as $\frac{w_{\text{epoch final}}}{w_{\text{epoch 1}}}$. In Figure 7, the weights for a subset of these variables are depicted across all epochs. It is important to keep in mind that using AdamW naturally leads to weight decay. Hence, a gradual decrease in weight values may not be of interest, whereas variables whose weights either decreased dramatically (suggesting they were particularly unhelpful) or conversely increased significantly can be deemed more informative.

For instance, the absolute weights associated with categories such as "sex_Prefer not to answer", "cocaine_past_90_days_Yes" or "gender_identity_Non-Binary/ENBY" saw the largest drop in their weight values during training (Figure 7). These variables were also found to be uninformative during our Exploratory Data Analysis(EDA), most likely because those categories contained very few observations. As a result, it is highly likely that any feature importance technique would find them unimportant, but once more data is collected, these variables can become useful.

The variable "alcohol_past_90_days_Yes" was found to be the most important variable by far. This tendency was expectedly reflected in the data: the rate of anxiety among people who consumed alcohol was roughly 28%, compared to 24% for those who did not.

We then compared the feature importance in the form of AAW of the first layer to SHapley Additive exPlanations (SHAP) [27].

Overall, the correlation between AAW and absolute SHAP values was relatively weak, around 0.353. This can be seen in Figure 8c. However, in Figures 8a and 8b, which show the 10 most important variables based on AAW and SHAP, respectively, 5 out of the 10 variables are the same, suggesting that, to an extent, AAW-based importance gets closer to what the SHAP values indicate once the most predictive variables are chosen. If the weight decay had been set to a larger value, it is possible that the weights of less valuable features would have decreased even more, bridging the gap between SHAP values and AAW. But unlike SHAP, AAW has a set of limitations that make them inferior as a form of feature importance. Yet, they can still be used to better understand how a given model functions.

5.2 Model Development Using a Subset of Variables

Initially, we used all available variables to train the models, giving them full access to a wide range of features. However, not all variables are typically collected in clinical settings, and collecting some variables can raise privacy concerns.

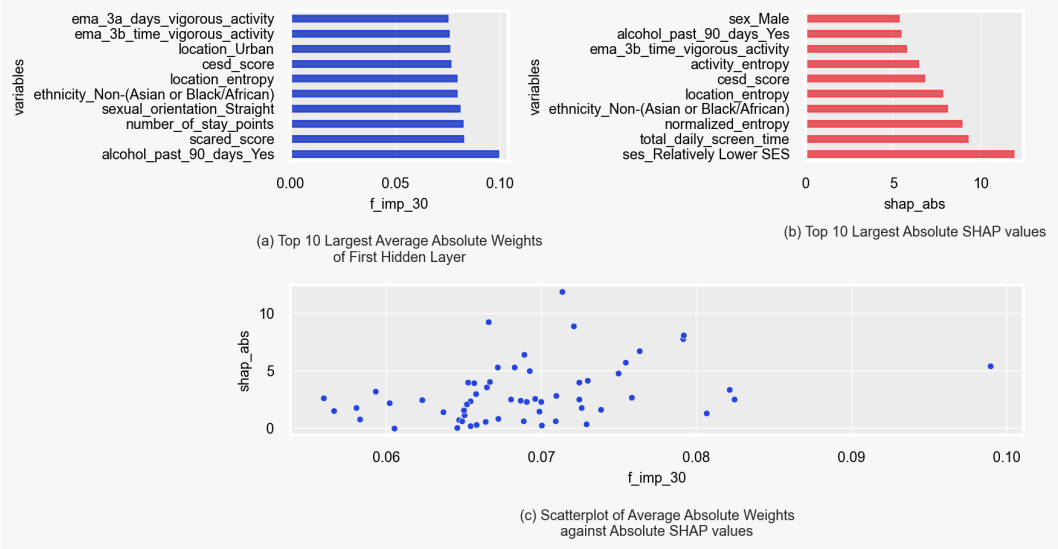


Figure 8: Comparing Average Absolute Weights (AAW) of First Layer to Absolute SHAP Values. Figures (a) and (b) show a significant overlap in the most important features derived from SHAP and AAW, whereas Figure (c) indicates that the overall correlation (considering all variables) between SHAP and AAW is relatively weak, at roughly 0.353.

As a result, we considered excluding all demographic variables (e.g., sex, age, education, ethnicity), sensitive variables such as drug use, and self-reported psychological assessments, reducing the total number of variables from 55 to 29.

Following that, we re-ran all shallow ML models along with our best DNN variant. The results can be found in Table 3.

Model	Score	F1 Class 0	F1 Class 1	F1 Macro	Balanced Accuracy	AUC
LR	avg	0.489	0.711	0.600	0.644	0.691
	std	0.021	0.032	0.024	0.020	0.014
RF	avg	0.371	0.841	0.606	0.598	0.731
	std	0.046	0.010	0.025	0.014	0.014
XGB	avg	0.420	**0.855**	0.637	0.624	**0.782**
	std	0.050	0.009	0.028	0.022	0.018
DNN	avg	**0.576**	0.825	**0.701**	**0.717**	0.778
	std	0.027	0.011	0.017	0.021	0.018

Table 3: Aggregated model performance scores over 15 runs (subset of 29 variables).

Class 0 denotes the minority class, representing instances of anxiety. When comparing the results to models trained with all features (Table 2), it is evident that the performance dropped substantially.

The DNN model remained the most accurate, albeit with a slightly lower AUC score than XGBoost. However, the removal of the aforementioned variables led to a significant drop in performance across all models, particularly for the minority class, demonstrating that the removed variables have high predictive power. This finding highlights the trade-off between maintaining high model performance and adhering to practical and ethical constraints, such as privacy preservation and feasibility in clinical settings.

6 Limitations

The key limitation of our study is the small sample size. Although supplementing it with synthetic data allowed us to overcome the issue of overfitting and achieve reasonable performance on the test set, the results obtained are most likely not generalizable beyond this dataset, as considerably more data are needed to properly model anxiety in the general population. Additionally, we exclusively used weighted loss functions to address the data imbalance problem. Other techniques could be explored in the future. Lastly, exploring a different approach to producing a train-test split should be considered, particularly one that keeps only unique participants in each split, as this would help ensure that the model's performance is not overly influenced by repeated data from the same participants.

7 Conclusion

We believe this study serves as a proof of concept, showcasing the advantages of using machine learning to predict anxiety. Most notably, while being accurate, ML models can learn the priors from the data, largely eliminating the problem of the posterior probability being negatively impacted by data imbalance, which is often the case with traditional medical tests.

Moreover, DL models can provide accurate probability estimates, which is particularly important in healthcare settings, as it enables healthcare providers to prioritise treatment in the context of limited resources.

Additionally, we demonstrated the advantages of using CTGANs for making tabular datasets more diverse, which can be highly beneficial in preventing models from overfitting. In theory, these benefits are not limited to small datasets and can be utilised for training networks on tabular data in general.

References

[1] Borwin Bandelow and Sophie Michaelis. Epidemiology of anxiety disorders in the 21st century. *Dialogues in Clinical Neuroscience*, 17(3):327–335, Sep 2015.

[2] Ronald C. Kessler, Hans M. van Loo, Klaas J. Wardenaar, Robert M. Bossarte, Laura A. Brenner, Tao Cai, David D. Ebert, I. Hwang, J. Li, P. de Jonge, Alan A. Nierenberg, Maria V. Petukhova, Ashley J. Rosellini, Naomi A. Sampson, Raoul A. Schoevers, Margaret A. Wilcox, and Alan M. Zaslavsky. Testing a machine-learning algorithm to predict the persistence and severity of major depressive disorder from baseline self-reports. *Molecular Psychiatry*, 21(10):1366–1371, October 2016.

[3] Arkaprabha Sau and Ishita Bhakta. Screening of anxiety and depression among the seafarers using machine learning technology. *Informatics in Medicine Unlocked*, 16:100149, 2019.

[4] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, October 2001.

[5] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *arXiv:1907.00503*, 2019.

[6] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks, 2018.

[7] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning, 2017.

[8] C. K. Chow and Chao-Ming Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theory*, 14:462–467, 1968.

[9] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via bayesian networks. *ACM Trans. Database Syst.*, 42(4), October 2017.

[10] Mikel Hernandez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Synthetic data generation for tabular health records: A systematic review. *Neurocomputing*, 493:28–45, 2022.

[11] W. Chen, K. Yang, Z. Yu, et al. A survey on imbalanced learning: latest research, applications and future directions. *Artificial Intelligence Review*, 57:137, May 2024. Accepted on 07 April 2024.

[12] Salim Rezvani and Xizhao Wang. A broad review on class imbalance learning techniques. *Applied Soft Computing*, 143:110415, 2023.

[13] Ravid Shwartz-Ziv and Amit Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[14] Wydmański Witold, Bulenok Oleksii, and Śmieja Marek. Hypertab: Hypernetwork approach for deep learning on small tabular datasets. *arXiv preprint arXiv:2304.03397*, 2023. Submitted on 7 Apr 2023, last revised 24 Aug 2023 (this version, v2).

[15] Sandra Meier. Prosit app, 2024. A research study aiming to improve the mental well-being of youth.

[16] Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67, 2011.

[17] Da Xu, Paul Jen-Hwa Hu, Ting-Shuo Huang, Xiao Fang, and Chih-Chin Hsu. A deep learning–based, unsupervised method to impute missing values in electronic health records for improved patient management. *Journal of Biomedical Informatics*, 111:103576, 2020.

[18] W. Dong, D.Y.T. Fong, J.S. Yoon, et al. Generative adversarial networks for imputing missing data for big data clinical research. *BMC Medical Research Methodology*, 21(1):78, 2021.

[19] Yige Sun, Jing Li, Yifan Xu, Tingting Zhang, and Xiaofeng Wang. Deep learning versus conventional methods for missing data imputation: A review and comparative study. *Expert Systems with Applications*, 227:120201, 2023.

[20] Yotam Elor and Hadar Averbuch-Elor. To smote, or not to smote? *arXiv preprint arXiv:2201.08528*, 2022. Version 3, last revised 11 May 2022.

[21] Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687, May 2021.

[22] Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits. *Transactions on Machine Learning Research*, 2024.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[24] Xiang Li, Shuo Chen, Xiaolin Hu, and Jian Yang. Understanding the disharmony between dropout and batch normalization by variance shift. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2677–2685, 2019.

[25] Olivia Remes, Carol Brayne, Rianne van der Linde, and Louise Lafortune. A systematic review of reviews on the prevalence of anxiety disorders in adult populations. *Brain and Behavior*, 6(7):e00497, 2016.

[26] M. Saerens, P. Latinne, and C. Decaestecker. Any reasonable cost function can be used for a posteriori probability approximation. *IEEE Transactions on Neural Networks*, 13(5):1204–1210, 2002.

[27] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

A Hyperparameter Grids

DNN:

n_hidden_layers: 1, 2, 3
dropout: 0.3, 0.25, 0.2, 0.1
weight_decay: 1e-4
epochs: 500
hidden_dim: 16, 32, 64
learning_rate: 1e-3
batch_size: 64
early_stopping_tol: 20

TabNet:

n_d: 4, 8, 16
n_steps: 2, 3
learning_rate: 1e-3
batch_size: 32
n_a: 4, 8, 16
cat_emb_dim: 4
weight_decay: 1e-4
epochs: 50, 100, 150

HyperTab:

hidden_dim: 0.5
subsample: 0.5
batch_size: 32
test_nodes: 100, 200, 300
learning_rate: 1e-3
epochs: 50, 100, 150

Random Forest (RF):

n_estimators: 50, 70, 100, 120, 150
min_samples_split: 5, 10, 15, 20, 30
bootstrap: True
criterion: 'log_loss'
max_depth: None, 10, 15, 20, 30
min_samples_leaf: 4, 8, 10, 15
class_weight: 'balanced'
max_features: None

XGBoost (XGB):

max_depth: 3, 5, 7, 9
subsample: 0.6, 0.8, 0.9
learning_rate: 1e-2
epochs: 100000
min_child_weight: 2, 5, 10, 15
colsample_bytree: 0.6, 0.8, 0.9
reg_lambda: 0.1, 1, 10
early_stopping_tol: 5